Tijdschrift voor Economie en Management
Vol. L, 4, 2005

From Conceptual Probabilistic Behavior to Operational Measurements

by G. DEDENE

Guido Dedene
KULeuven, Faculty of Economics and Management and Universiteit van Amsterdam Business School, Amsterdam, Nederland

ABSTRACT

This paper discusses a new approach to incorporate probabilistic behavior in object-oriented Business Models. Probabilistic Finite State Machines are presented and analyzed in terms of their Dynamic and Static properties. In this way, conceptual behavior models are translated in a precise fashion into measures that are very relevant from an operational point of view, for the implementation of the conceptual models. The techniques proposed here only represent a first step into a new area of Business Modeling.

I. INTRODUCTION

Traditionally, Object-oriented Conceptual Business Models, such as developed in MERODE (Snoeck and Dedene (1998) and Snoeck et al (1999)) and various UML-based formalisms, represent the behavior of objects in a fairly deterministic way. This paper will explore some possibilities to extend these behavior models by including probabilities in the Finite State Machines (FSM) that describe the internal behavior of objects.

This paper is written in honor of Em. Prof. Dr. J. Vandenbulcke, who was active particularly in Production, Inventory & Project Management (in the Belgian chapter of PICS) and – par excellence – in Database management (Vandenbulcke (2005)). The approach that is proposed in this paper is somewhat inspired by "activity on the node" versus "activity on the arrow" project models. It is intended as an exploration of new ways to extend the current frameworks for modeling the internal behavior of objects.

The paper starts with a precise definition of the type of probabilistic FSM that will be analysed and discussed. Many types of probabilistic FSM have been studied in literature before, in particular in Language Studies (Rabiner (1989)), Computer Performance Modeling and in Artificial Intelligence. Most of the result which have been developed there are not straightforward applicable to Business Objects. Hence some analysis refinements are proposed in the next paragraphs. First the dynamic behavior is discussed, focusing on everything that is related to events in the object lifecycles. A very interesting result is the determination of average lifecycle length. Next the static behavior is investigated, focusing on the static relationships which exist in models. These results are particularly interesting for database operational measures. The paper concludes with a discussion with suggestions for further research.

II. PROBABILISTIC BEHAVIOR IN CONCEPTUAL BUSINESS MODELS

As this paper tries to explore some new ideas, initially a basic example will be used to present these ideas. The apparent simplicity of this example should not mislead the reader in the applicability of the same type of analysis for larger systems.

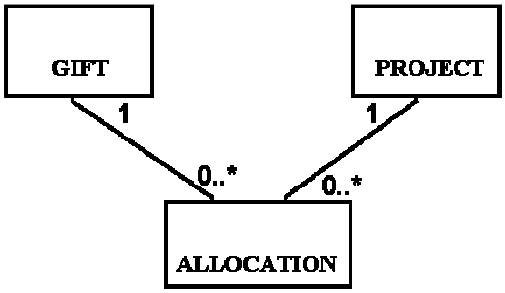
Consider the following specification for a fragment of a Business Model, adapted from a reference work on object-oriented Business Models (Snoeck et al (1999)):

> *A funding organization for supporting projects receives gifts. Gifts can be allocated in total or partially to projects. Gifts should be certified (once, and only once) for tax administration reasons. Projects have a*

description and a budget, which may be changed from time to time. All elements should be archived, but allocated parts of gifts cannot be archived unless the gift itself is certified

.

This model is somewhat inspired also by a real-life example (www.cfp.be). A traditional conceptual model for this Business description would use the following Class-Association diagram (in UML-notation format):



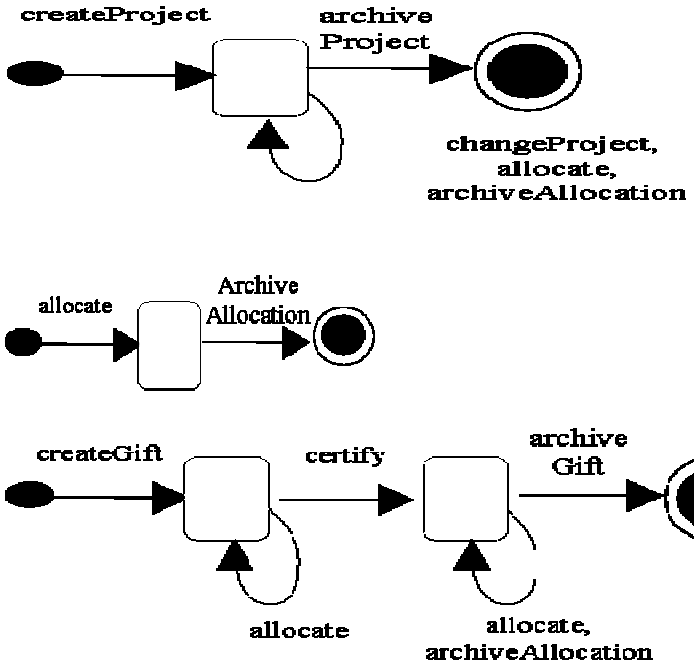
The relationships express that a gift may be allocated (partially) to many projects, and projects may receive money from various gifts. The Business events (which can also be considered as the Elementary Business Use Cases) should be consistent with this diagram (Snoeck and Dedene (1998)), and results in the following Class-Event Association Table, which indicates which classes are participating to what types of events:

	GIFT	ALLOCATION	PROJECT
createGift	C		
certify	M		
archiveGift	E		
allocate	M	C	M
archiveAllocation	M	E	M
creatProject			C
changeProject			M
archiveProject			E

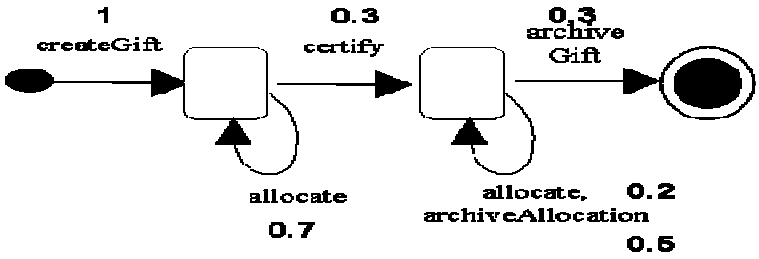
Each event has a number of participating object types, where a C indicates that a new object is created, a M denotes a (potential) modification of an object and an E indicates that this event terminates the life-cycle of an object.

From this table the life-cycle models follow in a straightforward fashion. Allocation and Project have trivial life-cycles: they are created and ended as show in the previous table. The life-cycle of a Gift is somewhat more complicated, due to the certification event, which may only happen once, and the particular situation on the archiving of Allocations.

Using stratified Finite State Machines, the life-cycles for the different Business objects are the following:



One way these lifecycle models can be improved with additional information is by means of adding probabilities to the state transitions. In first instance the lifecycle of Gift will be dressed with probabilities now. Suppose the following probabilities for the other state transitions are known:

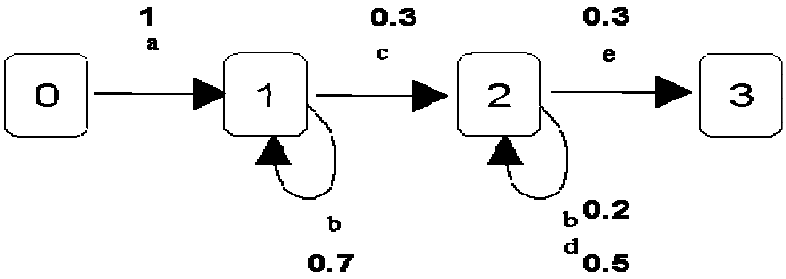


The probabilities should be interpreted as follows. In the first state, the probability of occurrence of a “certify”-event is 0.4 whereas the probability of an “allocate”-event is 0.6. It is clear that the sum of all outgoing probabilities should be equal to 1 for any given state. In this example the probabilities are fairly arbitrary although not unlogical, as will be seen clearly from other analysis results in this paper. It is clear that more “archiveAllocation”-events occur after the “certify”-event happened than before it happened. However the numbers have deliberately been kept simple, just to present the ideas.

The above probabilistic FSM can also be seen as a Markov chain, but it is quite different from the type of Markov chains that are used in Computer Performance Evaluation, for example. The main reason is the appearance of an explicit “start” state, which is actually not a really state, because it is the “nill” state before an object exists. So actually, an object is never in that state, but it is an explicit part of the FSM. Also Business Object lifecycles typically have a clear end-state. Many of the traditional Markov and Hidden Markov Chain analysis techniques are less applicable to this type of model. The next paragraphs will explore how operational measures can be derived from the above probabilistic FSM.

III. DYNAMIC OPERATIONAL MEASURES

The first type of analysis is related to the event behavior of the objects. In this way it can be seen as a dynamic operational view. For the sake of analysis, the FSM will be made somewhat more abstract, renaming the states and the events as follows:



A. *Steady State Probabilities*

If p_i denotes the steady state probability of being in a state i, and a_{ij} denotes the probability associated with the state transition from state i to state j in the FSM, the steady state probabilities are the solution of the following equations:

$$p_i = \sum_j p_j \times a_{ji} \qquad \text{(state balance equations)}$$

$$\sum_i p_i = 1 \qquad \text{(normalization condition)}$$

In this case the transition probability matrix is the following:

$$a_{ij} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0.7 & 0.3 & 0 \\ 0 & 0 & 0.7 & 0.3 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

The steady state solutions in this case are given by the following vector:

$$p = [0.115 \quad 0.385 \quad 0.385 \quad 0.115]$$

So far this is a classical Markov chain analysis. However in this case the state p_0 is not relevant since it is the state in which the object must still be created. The other states are the real "existing" states of the object Gift. Observe that the last state p_3 probability is equal to the first state p_0 probability, which is logical. It should also be clear that in a conceptual model an object that finished its lifecycle is ready to be deleted, but should not necessarily be deleted. This aspect will be covered again in the database related measures.

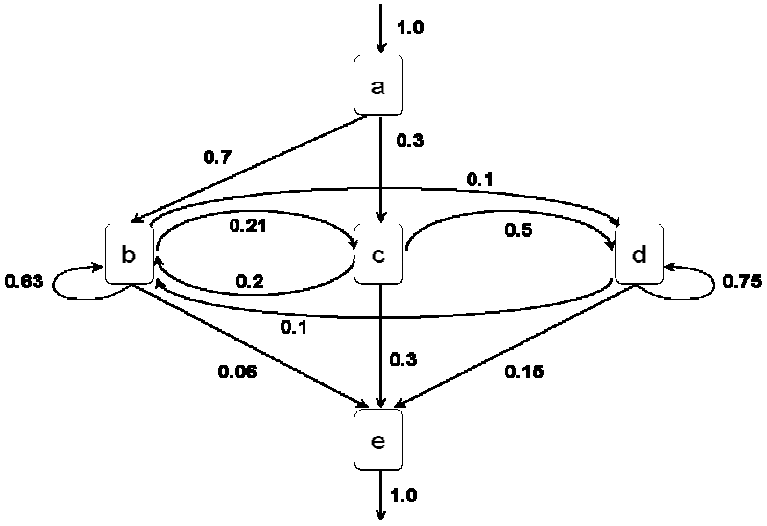
One solution consists of renormalizing the state probabilities for all the states different from the start state. In this case, the relevant state probabilities are given by:

$$p = [0.435 \quad 0.435 \quad 0.130]$$

which shows that there are as many Gifts certified as not certified yet.

A. *Event visit models*

The next kind of analysis requires a transformation of the probabilistic FSM into an event visit model, which shows the "events on the nodes" instead of "on the arrows" as it happens in an FSM. This model shows the probabilities of a visit to an event after a previous event. Using a straightforward heuristic (Rabiner (1989) and Menascé and Almeida (2000)), the following shows the transitions between the various events in the life-cycle of a Gift:



Observe how the outgoing state transition probabilities sum up to 1 for each event. The average number of visits to an event i, denoted by v_i is again equal to the sum of the number of visits to all states multiplied by the transition probability b_{ji} from each of the other states to state i. In this situation, the first event a is the entry event which has trivially one visit, and the terminating event e has no next state. So the set of equations to be solved is:

$$v_1 = 1$$

$$v_i = \sum_{j=1}^{n-1} v_j \times b_{ji} \qquad \text{i = 2,…,n-1}$$

if n denotes the total number of events. In this case this leads to the following average number of visits:

$$v = \begin{bmatrix} 1 & 3.333 & 1 & 3.333 & 1 \end{bmatrix}$$

Some elements are interesting and significant to notify:

The average number of visits to the certify (c) event is equal to 1, which is actually as to be expected, because every Gift can only be certified once.

The number of visits (per Gift) to the allocate (b) event is equal to the number of visits to the ArchiveAllocation (d) event. This is again very logical since every Allocation that is created once must also be archived once.

There are on average 3.333 allocations of (portions) of Gifts to Projects. Of course this is only an "average", but still an interesting observation.

A most interesting measure is the "average length", which gives the total number of visits to the events in the object lifecycle. In this case for a Gift it is equal to

$$L = \sum_i v_i = 9.666$$

So, on average, almost 10 events happen in the lifecycle of a Gift Business object.

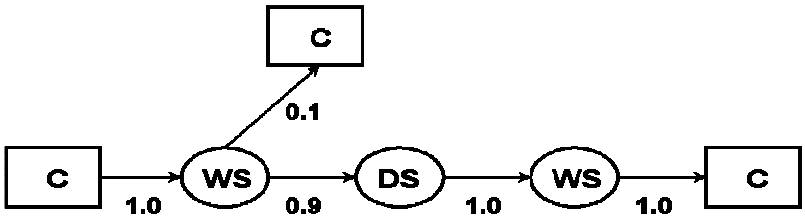
In other cases it can be interesting, for example, to get the buy-to-visit ratio's (assuming the life-cycle contains a "buy"-event) on the basis of the number of event visits.

C. *Resource usage models*

Once the event visit models have been analyzed, interesting capacity planning models can be developed. The events in the previous models are – by definition – Business events: they happen in the Business reality. In an activity-based approach to (service-oriented) capacity planning (Dedene et al (2004)), Business events should be translated into standard services. Suppose two standard services have been defined:

- Web Server (WS) : a Web Application Server.
- Database Server (DS) : a separate Database Server

Every Business event can be characterized by its Services Interaction Diagram (SID), which indicates the visit scenario's to the various standard services. Suppose the Allocation of a (portion) of a Gift has the following SID:



In this diagram C denotes the "client". In this case the Web Server rejects (on average) 10% of the Allocation request (purely for Internet rechnology reasons, other Business related scenario's must be modeled in the Business object lifecycle). The Database Server is activated on average 0.9 times, namely in the sequence C – WS – DS – WS – C . The Web Server is activated twice in the same sequence (with a probability 0.9) and one time in the other sequence, so the average number of times the Web Server is activated during an allocation = 2 x 0.9 + 1 x 0.1 = 1.9 times.

As a result there are for every Gift on average 1.9 x 3.33 = 6,33 activations of the Web Server. If it is know, for example, that typically 10,000 Gifts are created per month, the allocations alone will generate 63,300 activations of the Web Server (but maybe not necessarily during the same month). If the same analysis is performed for the other Business events, the total impact on the Web Server and Database Server services can be determined and translated into Service Demands on CPU and IO-devices.

IV. STATIC OPERATIONAL MEASURES

Another type of analysis may focus more on the objects themselves instead of the events. This can be seen as a more static operational view.

A. *Average cardinalities*

The Class-Association diagram indicates a one-to-many relationship between Gifts and Allocations. Many database resource management tools require database administrators to determine the actual average cardinalities. In this case, they follow from the probabilities that have been analyzed before: for each Gift there are on average 3.33 Allocations, which is in this case at least an upper bound for the average cardinalities.

Indeed, another factor that plays a role is the speed at which archived objects are actually removed from the database. If the speed is the same for all objects, that 3.33 would indeed be the average cardinality. If however, for some reason, Gift are deleted slower than Allocations, the average cardinality is indeed lower. However, referential integrity will require that a Gift cannot be deleted as long as there are Allocations that are dependent on it, which implies that typically Gifts will not be deleted faster than Allocations, in general.

B. *Database residence times*

The steady state probabilities also provide usable input for estimating the average database residence time for objects. In the case of Gifts, the objects may reside under 3 states in the database:

- Gifts that are created, but not yet certified
  (on average with a probability 0.435)
- Gifts that are certified but not yet archived
  (also with a probability 0.435)
- Gifts that are archived but not yet removed from the database
  (with a probability 0.130).

So the average probabilities under which we will find the objects in the database are known from this.

Another useful information is the average time between two relevant events. In this case, suppose it is known that

- The average time between creation of a Gift and its certification is 4 months.
- The average time between certification of a Gift and its archiving is 20 months.
- Suppose archived Gifts are kept still 24 months in the database before they are actually removed.

In this case this implies that every Gift resides on average 48 months in the database, in one of the 3 states that were mentioned before. A reverse engineering of the actual states of the objects in the database may help to refine the probabilistic behavior models

V. DISCUSSION AND FURTHER RESEARCH

This paper was only a first exploration of some extensions to current object-oriented conceptual modeling techniques to incorporate probabilistic behavior

in the models. For sure this type of extension is needed to build realistic Business Agents, i.e. active objects for which probabilistic behavior is evident.

Many topics proposed here require further research, such as:

- Simple algorithms for the translation of Probabilistic FSMs into Event Visit models, and the other way around.
- So far, only "correct" probabilistic behavior was included, leading – for example – to only one and exactly one "certify" event per Gift. In reality, errors can happens, and erroneous multiple certifications may happens are events, although they must be intercepted by the Business model implementation. Including this aspects will further refine the SIDs for the events.
- Algorithms for the extraction of the probabilities and/or the event visits from actual user behavior.
- The time series aspects of the models need a detailed further investigation. In particular bursting phenomena on some events may destroy completely the meaning of "average" results.
- The detailed translation into capacity planning models, elaborating more in detail some models of (Menascé et al (2000)).
- The application of the same results in a distributed Service-Oriented Architecture (Lemahieu et al (2005))

REFERENCES

Snoeck M., Dedene G., 1998, Existence Dependency: the Key to Semantic Integrity between Structural and Behavioral Aspects of Object Types, *IEEE Transactions on Software Engineering* 24, 4.

Snoeck M., Dedene G., Verhelst M. Depuydt A., 1999, Object-Oriented Enterprise Modelling with MERODE, (Leuven University Press).

Vandenbulcke J.A., Lemahieu W.O., 2005, Databasesystemen voor de praktijk, (ten Hagen & Stam Uitgevers).

Rabiner L.R., 1989, A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition, *Proceedings of the IEEE* 77, 2.

Menascé D.A., Almeida V.A., 2000, Scaling for E-Business: Technologies, Models, Performance and Capacity Planning, (Prentice Hall PTR Upper Saddle River, NJ).

Dedene G., Viaene S., Cumps B., De Backer M., 2004, An ABC-Based Approach for Operational Business – ICT alignment, Proceedings of the 11th European Conference on Information Technology Evaluation, (Amsterdam, The Netherlands).

Lemahieu W., Snoeck M., Goethals F., De Backer M., Haesen R., Dedene G., Vandenbulcke J., 2005, Coordinating COTS Applications by Means of a Business Event Layer, *IEEE Software* 22; 4.